A dynamic logic for every season*

Alexandre Madeira[1], Renato Neves[1], Manuel A. Martins[2] and Luís S. Barbosa[1]

[1] HASLab INESC TEC & Univ. Minho
{amadeira, renato.j.neves, luis.s.barbosa}@inesctec.pt
[2] CIDMA - Dep. Mathematics, Univ. Aveiro
martins@ua.pt

Abstract. This paper introduces a method to build dynamic logics with a graded semantics. The construction is parametrized by a structure to support both the spaces of truth and of the domain of computations. Possible instantiations of the method range from classical (assertional) dynamic logic to less common graded logics suitable to deal with programs whose transitional semantics exhibits fuzzy or weighted behaviour. This leads to the systematic derivation of program logics tailored to specific program classes

1 Introduction

Propositions, capturing static properties of program states, and events, or actions, which are responsible for transitions from a state to another, are the key ingredients in modelling and reasoning about state-based software systems. The latter are typically combined through a Kleene algebra to express sequential, non deterministic, iterative behaviour of systems, while the former brings to the scene a logical structure.

Dynamic logic [6], a generalisation of the logic of Floyd-Hoare, is a well known and particularly powerful way of combining these two dimensions into a formal framework to reason about computational systems. Its potential stems from blending together classical logic, enriched with a modal dimension to express system's dynamics, and a (Kleene) algebra of actions to structure programs.

Over time dynamic logic grew to an entire family of logics increasingly popular in the verification of computational systems, and able to evolve and adapt to new, and complex validation challenges. One could mention its role in model validation (as in *e.g.* [10]), or the whole family of variants tailored to specific programming languages (as in *e.g.* [11, 1]), or its important extensions to new computing domains, namely probabilistic [8] or continuous [13, 14].

* Accepted authors' manuscript published as: *A dynamic logic for every season*, Alexandre Madeira, Renato Neves, Manuel A. Martins and Luís S. Barbosa, edited by Christiano Braga, Narciso Martí-Oliet, Lecture notes in computer science, Volume 8941, 130–145, Springer International Publishing, 2015 [DOI:10.1007/978-3-319-15075-8_9]. The final publication is available at Springer via http://link.springer.com/chapter/10.1007%2F978-3-319-15075-8_9.

The latter is particularly relevant from an Engineering point of view: Actually, Platzer's hybrid dynamic logic, and its associated tool, KeYmaera, combining an algebra of actions based on real numbers assignments, with the standard Kleene operators and differential equations to specify continuous transitions from the "real" (physical) world, provides a powerful framework for the design and validation of cyber-physical systems with increased industrial relevance.

If cyber-physical systems gives rise to the need for ways of dealing with continuous state spaces, in a number of other cases dealing with some form of "quantitative" transitions (weighted, probabilistic, etc) is also a must. Hence the quest for dynamic logics able to capture smoothly these kind of phenomena is becoming more and more important.

This paper intends to contribute in this path. In particular, our attention is focussed on graded logics [4, 16], in the broad sense of attaching partially ordered grades to logical formulas to express, in one way or another, uncertain information. In this broad sense, fuzzy [5], probabilistic [12] or weighted logics [2] may be brought into the picture.

In this context, the purpose of this work is the development of a generic method to construct graded dynamic logics. Technically, the definition of these logics is parametrized by (a specific kind of) an action lattice [7] which combines (a slight generalisation of) an Kleene algebra with a residuated lattice structure. The latter captures the graded logic dimension and fits nicely with our objectives. Moreover, the extension of Kleene algebras with residuation operators, providing weak right and left inverses to sequential composition as in [15], as well as with a lattice structure leads to a finitely-based equational variety which, as plain Kleene algebras, is closed under the formation of square matrices [9]. The relevance of this closure property lies in the fact that several problems modelled as (weighted) transition systems can be formulated as matrices over a Kleene algebra or a related structure. Following such a trend, we represent programs as matrices supporting the information about their effects when executed from each state in the state space. The interested reader is referred to [3] for a detailed discussion on the relationship between Kleene algebras, action algebras and action lattices.

The remaining of this paper is organised as follows. Section 2 recalls from [7] the definition of an action lattice and introduces a method, parametric on such a lattice, to generate graded dynamic logics. The construction put forward is illustrated with several examples. Then, in Section 3, it is shown that the resulting logic is a dynamic logic indeed, in the sense that all the rules of propositional dynamic logic restricted to positive-existential formulas still hold. Finally, Section 4 concludes and suggests points for future research.

2 The method

This section introduces a generic method to generate *graded dynamic logics* parametric on a complete action lattice which captures both the structure of the computational domain and that of the (logical) truth space.

Let us start by recalling from [7] the following definition:

Definition 1. *An* action lattice *is a tuple*

$$\mathbf{A} = (A, +, ;, 0, 1, *, \leftarrow, \rightarrow, \cdot)$$

where A is a set, 0 and 1 are constants and $+, ;, *, \leftarrow, \rightarrow$ and $\cdot$ are binary operations in A satisfying the axioms in Figure 1, where the relation $\leq$ is the one induced by $+$ as $a \leq b$ iff $a + b = b$.

$$\begin{aligned}
a + (b + c) &= (a + b) + c && (1)\\
a + b &= b + a && (2)\\
a + a &= a && (3)\\
a + 0 &= 0 + a = a && (4)\\
a; (b; c) &= (a; b); c && (5)\\
a; 1 &= 1; a = a && (6)\\
a; (b + c) &= (a; b) + (a; c) && (7)\\
(a + b); c &= (a; c) + (b; c) && (8)\\
a; 0 &= 0; a = 0 && (9)\\
1 + a + (a^*; a^*) &\leq a^* && (10)\\
a; x \leq x &\Rightarrow a^*; x \leq x && (11)\\
x; a \leq x &\Rightarrow x; a^* \leq x && (12)\\
a; x \leq b &\Leftrightarrow x \leq a \rightarrow b && (13)\\
x; a \leq b &\Leftrightarrow x \leq a \leftarrow b && (14)\\
(x \rightarrow x)^* &= x \rightarrow x && (15)\\
(x \leftarrow x)^* &= x \leftarrow x && (16)\\
a \cdot (b \cdot c) &= (a \cdot b) \cdot c && (17)\\
a \cdot b &= b \cdot a && (18)\\
a \cdot a &= a && (19)\\
a + (a \cdot b) &= a && (20)\\
a \cdot (a + b) &= a && (21)
\end{aligned}$$

Fig. 1. Axiomatisation of action lattices (from [7])

An action lattice is said to be *complete* when there are both a supremum and an infimum, wrt $\leq$, of all subsets of A. Therefore, complete action lattices have biggest and smallest elements denoted in the sequel by $\top$ and $\bot$, respectively. Note that in any action lattice $\bot = 0$, since for any $a \in A$, $a+0 = a$, i.e., $0 \leq a$. In this paper we resort to notation $\sum$ for the iterated version of the (join) operator $+$, and to notation $\prod$ for the iterated version of the (meet) operator $\cdot$.

The starting point for the method proposed here is thus the choice of an appropriate complete action lattice

$$\mathbf{A} = (A, +, ;, 0, 1, *, \leftarrow, \rightarrow, \cdot)$$

Additionally, we require $\mathbf{A}$ to satisfies the following distributive law:

$$a; (b \cdot c) = a; b \cdot a; c \tag{22}$$

As mentioned above, this structure supports both the computational paradigm (to distinguish between *e.g.* imperative, deterministic or non deterministic computations, or between plain or weighted transitions) and the truth space (to capture *e.g.* the standard Boolean reasoning or more complex truth spaces). Before proceeding let us exemplify this structure with a couple of action lattices typically found in Computer Science applications. In the examples, the logic generated by an action lattice $\mathbf{A}$ will be denoted by $\mathcal{GDL}(\mathbf{A})$.

Example 1 ($\mathcal{GDL}(\mathbf{2})$ — the standard propositional dynamic logic). Standard propositional dynamic logic is generated from the following structure

$$\mathbf{2} = (\{\top, \bot\}, \vee, \wedge, \bot, \top, *, \leftarrow, \rightarrow, \wedge)$$

with the standard boolean connectives:

$\vee$	$\bot$	$\top$
$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$

$\wedge$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\top$

$\rightarrow$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$

$*$	
$\bot$	$\top$
$\top$	$\top$

and taking $a \leftarrow b = b \rightarrow a$. It is not difficult to see that $\mathbf{2}$ is an action algebra. Moreover, the lattice is obviously complete and it satisfy the condition (22) (note that both composition and the meet operator are realized by $\wedge$).

Example 2 ($\mathcal{GDL}(\mathbf{3})$ — a dynamic logic to deal with unknown data). This is a three-valued logic, with an explicit representative for *unknown*, or *uncertain* information. Note that the three elements linear lattice induces an action lattice

$$\mathbf{3} = (\{\top, u, \bot\}, \vee, \wedge, \bot, \top, *, \leftarrow, , \rightarrow, \wedge)$$

where

$\vee$	$\bot$	u	$\top$
$\bot$	$\bot$	u	$\top$
u	u	u	$\top$
$\top$	$\top$	$\top$	$\top$

$\wedge$	$\bot$	u	$\top$
$\bot$	$\bot$	$\bot$	$\bot$
u	$\bot$	u	u
$\top$	$\bot$	u	$\top$

$\rightarrow$	$\bot$	u	$\top$
$\bot$	$\top$	$\top$	$\top$
u	$\bot$	$\top$	$\top$
$\top$	$\bot$	u	$\top$

$*$	
$\bot$	$\top$
u	$\top$
$\top$	$\top$

and taking $a \leftarrow b = b \rightarrow a$. It is easy to see all the conditions in Definition 1 hold. Moreover, the lattice is complete and satisfies condition (22). The reader should note that both composition and meet are realized by $\wedge$).

Example 3 ($\mathcal{GDL}(Ł)$ — a dynamic logic to deal with continuous levels of fuzziness).

This is based on the well-known Łukasiewicz arithmetic lattice

$$Ł = ([0,1], max, \odot, 0, 1, *, \rightarrow, \leftarrow, min)$$

where

- $x \odot y = max\{0, y + x - 1\}$
- $x \rightarrow y \;=\; min\{1, 1 - x + y\}$,
- $x \leftarrow y \;=\; 1 - max\{0, x + y - 1)\}$ and
- $*$ maps each point of $[0,1]$ to 1.

Again, this defines a complete action lattice which additionally satisfies condition (22). Note that both composition and the meet operator are now represented by function min.

Example 4 ($\mathcal{GDL}(FW)$ – a dynamic logic to deal with resource consuming systems). This example explores the so called Floyd-Warshall algebra which consists of a tuple $\mathbb{N}^{+}_{\bot\top} = (\{\bot, 0, 1, \ldots, \top\}, max, +, \bot, 0, *, \curvearrowright, \curvearrowleft, min)$ where $+$ extends the addition on $\mathbb{N}$ by considering $\bot$ as its absorbent element and $a + \top = \top = \top + a$ for any $a \neq \bot$. The operation max (and min) are defined as the maximum (minimum) under the order $\bot \leq 0 \leq \cdots \leq \top$. The operation $\curvearrowright$ is the truncated subtraction

$$a \curvearrowright b = \begin{cases} \top, & \text{if } a = \bot \text{ or } b = \top \\ b - a, & \text{if } b \geq a \text{ and } a, b \in \mathbb{N} \\ 0, & \text{if } a > b \text{ and } a, b \in \mathbb{N} \\ \bot & \text{otherwise} \end{cases}, \qquad a \curvearrowleft b = b \curvearrowright a \quad \text{and}$$

for any natural $i > 0$,

$*$	
$\bot$	0
0	0
i	$\top$
$\top$	$\top$

. Note that the order induced by $a \leq b$ iff $max\{a, b\} = b$ corresponds to the mentioned above. The lattice is also complete and it satisfies condition (22) because $a + min\{b, c\} = min\{a + b, a + c\}$.

Illustrated the notion of an action lattice, we are now prepared to introduce the general construction of graded dynamic logics. We consider now its signatures, formulæ, semantics and satisfaction. Thus,

Signatures. Signatures of $\mathcal{GDL}(\mathbf{A})$ are pairs (Π, Prop) corresponding to the denotations of atomic computations and of propositions, respectively.

Formulæ. A core ingredient of any dynamic logic is its set of programs. Therefore, let us denote the set of atomic programs by Π. The *set of Π-programs*, denoted by $\mathrm{Prg}(\Pi)$, consists of all expressions generated by

$$\pi \ni \pi_0 \,|\, \pi; \pi \,|\, \pi + \pi \,|\, \pi^*$$

for $\pi_0 \in \Pi$. Given a signature (Π, Prop), we define the $\mathcal{GDL}(\mathbf{A})$-formulas for (Π, Prop), denoted by $\mathrm{Fm}^{\mathcal{GDL}(\mathbf{A})}(\Pi, \mathrm{Prop})$, by the grammar

$$\rho \ni \top \mid \bot \mid p \mid \rho \vee \rho \mid \rho \wedge \rho \mid \rho \to \rho \mid \langle \pi \rangle \rho$$

for $p \in \mathrm{Prop}$ and $\pi \in \mathrm{Prg}(\Pi)$. Note that this corresponds to the *positive existential* fragment of the propositional dynamic logic.

Semantics. The first step is to introduce the space where the computations of $\mathcal{GDL}(\mathbf{A})$ are to be interpreted. As usual, this corresponds to a Kleene algebra. Therefore, we consider the structure

$$\mathbb{M}_n(\mathbf{A}) = (M_n(\mathbf{A}), +, ;, \mathbf{0}, \mathbf{1}, ^*)$$

defined as follows:

1. $M_n(\mathbf{A})$ is the space of $(n \times n)$-matrices over $\mathbf{A}$
2. for any $A, B \in M_n(\mathbf{A})$, define $M = A + B$ by $M_{i,j} = A_{i,j} + B_{i,j}$, $i, j \leq n$.
3. for any $A, B \in M_n(\mathbf{A})$, define $M = A \,;\, B$ by taking $M_{i,j} = \sum_{k=1}^{n} (A_{i,k}; B_{k,j})$ for any $i, j \leq n$.
4. $\mathbf{1}$ and $\mathbf{0}$ are the $(n \times n)$-matrices defined by $\mathbf{1}_{i,i} = 1$ and $\mathbf{0}_{i,j} = 0$, for any $i, j \leq n$.
5. for any $M = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \in M_n(\mathbf{A})$, where A and D are square matrices, we take

$$M^* = \left[\begin{array}{c|c} F^* & F^* \,;\, B \,;\, D^* \\ \hline D^* C F^* & D^* + D^* \,;\, C \,;\, F^* \,;\, B \,;\, D^* \end{array}\right]$$

   where $F = A + B \,;\, D^* \,;\, C$. Note that this construction is recursively defined from the base case (where $n = 2$) where the operations of the base action lattice $\mathbf{A}$ are used.

Finally, we have to show that,

Theorem 1. *The structure* $\mathbb{M}_n(\mathbf{A}) = (M_n(\mathbf{A}), +, ;, \mathbf{0}, \mathbf{1}, *)$ *defined above is a Kleene algebra.*

Proof. The structure, and the respective operations, corresponds to the algebra of matrices over $(A, +, ;, 0, 1, *)$, *i.e.*, the Kleene algebra underlying action lattice $\mathbf{A}$. A canonical result establishes that Kleene algebras are closed under formation of matrices (e.g. [9]). Therefore, $\mathbb{M}_n(\mathbf{A})$ constitutes a Kleene algebra. □

$\mathcal{GDL}(\mathbf{A})$-models for a set of propositions Prop and programs Π, denoted by $\mathrm{Mod}^{\mathcal{GDL}(\mathbf{A})}(\Pi, \mathrm{Prop})$, consists of tuples

$$\mathcal{A} = (W, V, (\mathcal{A}_\pi)_{\pi \in \Pi})$$

where

– W is a finite set (of states),

– $V : \mathrm{Prop} \times W \to A$ is a function,
– and $\mathcal{A}_\pi \in \mathbb{M}_n(\mathbf{A})$, with n standing for the cardinality of W.

The interpretation of programs in these models is made by matrices over the Kleene algebra of $\mathbf{A}$. Each matrix represents the effect of a program executing from any point of the model. Formally, the interpretation of a program $\pi \in \mathrm{Prg}(\Pi)$ in a model $\mathcal{A} \in \mathrm{Mod}^{\mathcal{GDL}(\mathbf{A})}(\Pi, \mathrm{Prop})$ is recursively defined, from the atomic programs $(\mathcal{A}_\pi)_{\pi \in \Pi}$, as follows:

– $\mathcal{A}_{\pi;\pi'} = \mathcal{A}_\pi \,;\, \mathcal{A}_{\pi'}$
– $\mathcal{A}_{\pi+\pi'} = \mathcal{A}_\pi + \mathcal{A}_{\pi'}$
– $\mathcal{A}_{\pi^*} = \mathcal{A}_\pi^*$

Observe that the set of states W supports the index system of the programs (adjacency) matrices. In this context, it is important to note, that, for example,

$$(M_{\pi;\pi'})_{ij} = \sum_{k=1}^{n} \{(M_\pi)_{ik}; (M_{\pi'})_{kj}\}$$

corresponds to

$$(M_{\pi;\pi'}(w, w') = \sum_{w'' \in W} \{(M_\pi)(w, w''); (M_{\pi'}(w'', w')\}$$

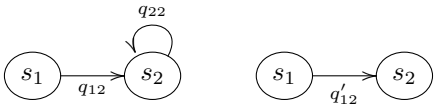
where i and j stands for the adjacency index of w and w', respectively. Actually, the latter characterisation is often used in the sequel.

Example 5 (Computations spaces).

Let us fix an action lattice $\mathbf{A} = (A, +, ;, 0, 1, *, \leftarrow, \rightarrow, \cdot)$ and a signature $(\{\pi, \pi'\}, \{p\})$. Then, consider a model $\mathcal{A} = (W, V, (\mathcal{A}_\pi)_{\pi \in \Pi})$, with $W = \{s_1, s_2\}$ and the following atomic programs

$$\mathcal{A}_\pi = \begin{bmatrix} \bot & q_{12} \\ \bot & q_{22} \end{bmatrix} \qquad \mathcal{A}_{\pi'} = \begin{bmatrix} \bot & q'_{12} \\ \bot & \bot \end{bmatrix}$$

which can be represented by the following labelled transition systems:



Let us suppose that $\mathbf{A}$ is realized by

$$\mathbf{2} = (\{\top, \bot\}, \vee, \wedge, \bot, \top, *, \leftarrow, \rightarrow, \wedge)$$

Making $q_{12} = q_{22} = q'_{1,2} = \top$ we get the standard adjacency matrices of the graph underlying the transition systems. In this case, we interpret choice $\pi + \pi'$ and composition $\pi; \pi'$ by

$$\mathcal{A}_{\pi+\pi'} = \mathcal{A}_\pi + \mathcal{A}_{\pi'} = \begin{bmatrix} \bot & \top \\ \bot & \top \end{bmatrix} + \begin{bmatrix} \bot & \top \\ \bot & \bot \end{bmatrix} = \begin{bmatrix} \bot \vee \bot & \top \vee \top \\ \bot \vee \bot & \top \vee \bot \end{bmatrix} = \begin{bmatrix} \bot & \top \\ \bot & \top \end{bmatrix}$$

The interpretation of the composition $\pi;\pi'$ is computed as follows,

$$\mathcal{A}_{\pi;\pi'} = \begin{bmatrix} \bot & \top \\ \bot & \top \end{bmatrix} ; \begin{bmatrix} \bot & \top \\ \bot & \bot \end{bmatrix} = \begin{bmatrix} (\bot \wedge \bot) \vee (\top \wedge \bot) & (\bot \wedge \top) \vee (\top \wedge \bot) \\ (\bot \wedge \bot) \vee (\top \wedge \bot) & (\bot \wedge \top) \vee (\top \wedge \bot) \end{bmatrix} = \begin{bmatrix} \bot & \bot \\ \bot & \bot \end{bmatrix}$$

As expected,

$$\mathcal{A}_{\pi';\pi} = \begin{bmatrix} \bot & \top \\ \bot & \bot \end{bmatrix}$$

For the interpretation of the π closure, we have

$$\mathcal{A}_{\pi^*} = (\mathcal{A}_{\pi})^* \begin{bmatrix} \bot & \top \\ \bot & \top \end{bmatrix}^* = \begin{bmatrix} f^* & f^* \wedge \top \wedge \top^* \\ \bot & \top^* \vee (\top^* \wedge \bot \wedge \top \wedge \top) \end{bmatrix}$$

where $f = \bot \vee (\top \wedge \top^* \wedge \bot) = \bot$; hence

$$\mathcal{A}_{\pi *} = \begin{bmatrix} \top & \top \\ \bot & \top \end{bmatrix}$$

as expected.

Taking the same matrix in the case of

$$\mathbf{3} = (\{\top, u, \bot\}, \vee, \wedge, \bot, \top, *, \leftarrow, \rightarrow, \wedge)$$

and considering $q_{12} = q_{22} = \top$ and $q'_{12} = u$, let us compute composition

$$\mathcal{A}_{\pi';\pi} = \begin{bmatrix} \bot & u \\ \bot & \bot \end{bmatrix} ; \begin{bmatrix} \bot & \top \\ \bot & \top \end{bmatrix} = \begin{bmatrix} (\bot \wedge \bot) \vee (u \wedge \bot) & (\bot \wedge \top) \vee (u \wedge \top) \\ (\bot \wedge \bot) \vee (\bot \wedge \bot) & (\bot \wedge \top) \vee (\bot \wedge \top) \end{bmatrix} = \begin{bmatrix} \bot & u \\ \bot & \bot \end{bmatrix}$$

As expected, the unknown factor affecting transition $s_1 \rightarrow s_2$ in $\mathcal{A}'_{\pi}$ is propagated to transition $s_2 \rightarrow s_2$ in $\mathcal{A}_{\pi';\pi}$.

If a continuous space is required to define the "unknown metric", one may resort to the Łukasiewicz arithmetic lattice

$$Ł = ([0,1], max, \odot, 0, 1, *, \rightarrow, \leftarrow, min)$$

Consider, for instance, $q_{12} = a$, $q_{22} = b$ and $q'_{12} = c$ for some $a, b, c \in [0,1]$. In this case we may, for example, compute choice $\pi + \pi'$, making

$$\mathcal{A}_{\pi+\pi'} = \begin{bmatrix} 0 & a \\ 0 & b \end{bmatrix} + \begin{bmatrix} 0 & c \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} max\{0,0\} & max\{a,c\} \\ max\{0,0\} & max\{b,0\} \end{bmatrix} = \begin{bmatrix} 0 & max\{a,c\} \\ 0 & b \end{bmatrix}$$

The reader may check that $\mathcal{A}_{\pi^*} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$. Note that the certainty value 1 in the diagonal of the matrix stands for the reflexive dimension of the reflexive-transitive closure $*$.

Let us now consider the action lattice

$$\mathbb{N}^+_{\bot\top} = (\{\bot, 0, 1, \ldots, \top\}, max, +, \bot, 0, *, \backsim, \backsim, min)$$

As stated above, the structure $\mathbb{N}^+_{\perp\top}$ is suitable to reason about resource consuming systems. The value of a transition is $\top$ when it costs an infinite amount of resources; it is $\perp$ when undefined. The composition of actions reflects the accumulation of sequential costs. For instance $\mathcal{A}_{\pi';\pi} =$

$$\begin{bmatrix} \perp & c \\ \perp & \perp \end{bmatrix} ; \begin{bmatrix} \perp & a \\ \perp & b \end{bmatrix} = \begin{bmatrix} max\{\perp+\perp, c+\perp\} & max\{\perp+a, c+b\} \\ max\{\perp+\perp, \perp+\perp\} & max\{\perp+a, \perp+b\} \end{bmatrix} = \begin{bmatrix} \perp & c+b \\ \perp & \perp \end{bmatrix}$$

Moreover, the interpretation of a program $\pi + \pi'$ reflects, in each transition the most expensive choice:

$$\mathcal{A}_{\pi'+\pi} = \begin{bmatrix} \perp & c \\ \perp & \perp \end{bmatrix} + \begin{bmatrix} \perp & a \\ \perp & b \end{bmatrix} = \begin{bmatrix} max\{\perp, \perp\} & max\{c, a\} \\ max\{\perp, \perp\} & max\{\perp, b\} \end{bmatrix} = \begin{bmatrix} \perp & max\{c, a\} \\ \perp & b \end{bmatrix}$$

Finally, observe the interpretation of the closure of π

$$\mathcal{A}_{\pi^*} = \begin{bmatrix} \perp & a \\ \perp & b \end{bmatrix} = \begin{bmatrix} f^* & f^* + a + b^* \\ \perp & max\{b^*, b^* + \perp + \perp^* + a + b^*\} \end{bmatrix} = \begin{bmatrix} 0 & a + b^* \\ \perp & b^* \end{bmatrix}$$

where $f = max\{\perp, a + b^* + \perp\}$. Note that for any $b > 0$, the matrix assumes $\begin{bmatrix} 0 & \top \\ \perp & \top \end{bmatrix}$ which reflects the cost of an undetermined repetition of transition $s_2 \to s_2$. Naturally, when the cost of the action is 0, we have $\begin{bmatrix} 0 & a \\ \perp & 0 \end{bmatrix}$.

Satisfaction. Finally, let us define the (graded) satisfaction relation. As mentioned above, the carrier of $\mathbf{A}$ corresponds to the space of truth degrees for $\mathcal{GDL}(\mathbf{A})$. Hence, the graded satisfaction relation for a model $\mathcal{A} \in \mathrm{Mod}^{\mathcal{GDL}(\mathbf{A})}(\Pi, \mathrm{Prop})$ consists of a function

$$\models : W \times \mathrm{Fm}^{\mathcal{GDL}(\mathbf{A})}(\Pi, \mathrm{Prop}) \to A$$

recursively defined as follows:

- $(w \models \top) = \top$
- $(w \models \perp) = \perp$
- $(w \models p) = V(p, w)$, for any $p \in \mathrm{Prop}$
- $(w \models \rho \wedge \rho') = (w \models \rho) \cdot (w \models \rho')$
- $(w \models \rho \vee \rho') = (w \models \rho) + (w \models \rho')$
- $(w \models \rho \to \rho') = (w \models \rho) \to (w \models \rho')$
- $(w \models \langle \pi \rangle \rho) = \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w' \models \rho)\}$

Example 6. In order to make a case for the versatility and generality of this method, let us consider the evaluation of the very simple sentence $\langle \pi^* \rangle p$ in three of the dynamic logics constructed in the examples above. Concretely, let us evaluate $\langle \pi^* \rangle p$ in state s_1. For this we calculate

$$(s_1 \models \langle \pi^* \rangle p) = \sum_{w' \in W} \{\mathcal{A}_{\pi^*}(s_1, w'); (w' \models p)\}$$

Starting with $\mathcal{GDL}(\mathbf{2})$, let us assume $V(p, s_1) = \bot$ and $V(p, s_2) = \top$. In this case, as expected

$$\begin{aligned}(s_1 \models \langle\pi^*\rangle p) &= \textstyle\sum_{w' \in W}\{\mathcal{A}_{\pi^*}(s_1, w'); (w' \models p)\}\\ &= \big(\mathcal{A}_{\pi^*}(s_1, s_1) \wedge (s_1 \models p)\big) \vee \big(\mathcal{A}_{\pi^*}(s_1, s_2) \wedge (s_2 \models p)\big)\\ &= (\top \wedge V(p, s_1)) \vee (\top \wedge V(p, s_2))\\ &= (\top \wedge \bot) \vee (\top \wedge \top)\\ &= \top\end{aligned}$$

This means that we can achieve p from s_1 through π^*.

Considering the $\mathcal{GDL}(\text{Ł})$ and assuming $V(s_1, p) = 0$ and $V(s_2, p) = 1$, we may calculate

$$\begin{aligned}(s_1 \models \langle\pi^*\rangle p) &= \textstyle\sum_{w' \in W}\{\mathcal{A}_{\pi^*}(s_1, w'); (w' \models p)\}\\ &= max\big\{min\{\mathcal{A}_{\pi^*}(s_1, s_1), (s_1 \models p)\},\\ &\qquad min\{\mathcal{A}_{\pi^*}(s_1, s_2), (s_2 \models p)\}\big\}\\ &= max\big\{min\{1, 0\}, min\{a, 1\}\big\}\\ &= max\{0, a\}\\ &= a\end{aligned}$$

Therefore, we can assure, with a degree of certainty a, that we can achieve p from s_1 through π^*.

Interpreting now the same sentence in logic $\mathcal{GDL}(\mathbb{N}^+_{\bot\top})$, assuming that $V(s_1, p) = \bot$ and $V(s_2, p) = 0$, we get

$$\begin{aligned}(s_1 \models \langle\pi^*\rangle p) &= \textstyle\sum_{w' \in W}\{\mathcal{A}_{\pi^*}(s_1, w'); (w' \models p)\big\}\\ &= max\big\{\mathcal{A}_{\pi^*}(s_1, s_1) + (s_1 \models p), \mathcal{A}_{\pi^*}(s_1, s_2) + (s_2 \models p)\big\}\\ &= max\{0 + \bot, a + b^* + 0\}\\ &= a + b^*\end{aligned}$$

Hence, we can say that p can be accessed from s_1 through π^* consuming $a + b^*$ resources unities.

3 "Dynamisations" are dynamic

Having introduced a generic method for generating dynamic logics, this section establishes that the resulting logics behave, in fact, as dynamic logics. In particular, all the axioms of the propositional dynamic logic involving positive-existential formulas (see [6]) remain sound in this generic construction.

In the context of graded satisfaction, the verification that a property ρ is valid corresponds to the verification that, for any state w of any model $\mathcal{A}$, $(w \models \rho) = \top$. Hence, assuming that A is integral (i.e., $1 = \top$), by (13) and (14), we have that asserting $(\rho \leftrightarrow \rho') = \top$ is equivalent to prove that, for any $w \in W$, $(w \models \rho) = (w \models \rho')$; and to proof $(\rho \to \rho') = \top$ is equivalent to proof that $(w \models \rho) \leq (w \models \rho')$.

Lemma 1. *The following are valid formulas in any* $\mathcal{GDL}(\mathbf{A})$*:*

(1.1) $\langle\pi\rangle(\rho \vee \rho') \leftrightarrow \langle\pi\rangle(\rho) \vee \langle\pi\rangle\rho'$
(1.2) $\langle\pi\rangle(\rho \wedge \rho') \to \langle\pi\rangle(\rho) \wedge \langle\pi\rangle\rho'$

Proof. **Axiom (1.1)**

$$
\begin{aligned}
& (w \models \langle\pi\rangle(\rho \vee \rho')) \\
= & \quad \{ \text{ defn of } \models \} \\
& \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w' \models \rho \vee \rho')\} \\
= & \quad \{ \text{ defn. of } \models \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); \big((w' \models \rho) + (w' \models \rho')\big)\} \\
= & \quad \{ \ (7) \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); (w' \models \rho) + (\mathcal{A}_\pi(w, w'); (w' \models \rho'))\} \\
= & \quad \{ \text{ supremum properties} \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); (w' \models \rho)\} + \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w' \models \rho'))\} \\
= & \quad \{ \text{ defn of } \models \} \\
& (w \models \langle\pi\rangle\rho) + (w \models \langle\pi\rangle\rho) \\
= & \quad \{ \text{ defn of } \models \} \\
& (w \models \langle\pi\rangle\rho \vee \langle\pi\rangle\rho)
\end{aligned}
$$

Therefore $\langle\pi\rangle(\rho \vee \rho') \leftrightarrow \langle\pi\rangle\rho \vee \langle\pi\rangle\rho$ is valid.

Axiom (1.2)

$$
\begin{aligned}
& (w \models \langle\pi\rangle(\rho \wedge \rho')) \\
= & \quad \{ \text{ defn of } \models \} \\
& \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w' \models \rho \wedge \rho')\} \\
= & \quad \{ \text{ defn. of } \models \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); \big((w' \models \rho) \cdot (w' \models \rho')\big)\} \\
= & \quad \{ \ (22) \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); (w' \models \rho) \cdot (\mathcal{A}_\pi(w, w'); (w' \models \rho'))\} \\
\leq & \quad \{ \text{ infimum properties} \} \\
& \sum_{w' \in W} \{(\mathcal{A}_\pi(w, w'); (w' \models \rho)\} \cdot \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w' \models \rho'))\}
\end{aligned}
$$

$$
\begin{aligned}
&= \quad \{ \text{ defn of } \models \} \\
&\quad (w \models \langle\pi\rangle\rho) \cdot (w \models \langle\pi\rangle\rho') \\
&= \quad \{ \text{ defn of } \models \} \\
&\quad (w \models \langle\pi\rangle\rho \wedge \langle\pi\rangle\rho')
\end{aligned}
$$

Therefore, $\langle\pi\rangle(\rho \wedge \rho') \rightarrow \langle\pi\rangle\rho \wedge \langle\pi\rangle\rho'$ is valid.

Lemma 2. *The following are valid formulas in any $\mathcal{GDL}(\mathbf{A})$:*

(2.1) $\langle\pi + \pi'\rangle\rho \leftrightarrow \langle\pi\rangle\rho \vee \langle\pi\rangle\rho$
(2.2) $\langle\pi; \pi'\rangle\rho \leftrightarrow \langle\pi\rangle\langle\pi'\rangle\rho$
(2.3) $\langle\pi\rangle\bot \leftrightarrow \bot$

Proof. **Axiom (2.1)**

$$
\begin{aligned}
&\quad (w \models \langle\pi + \pi'\rangle\rho) \\
&= \quad \{ \text{ defn of } \models \} \\
&\quad \sum_{w' \in W} \{\mathcal{A}_{\pi+\pi'}(w, w'); (w' \models \rho)\} \\
&= \quad \{ \text{ defn of programs interpretation} \} \\
&\quad \sum_{w' \in W} \{(\mathcal{A}_{\pi}(w, w') + \mathcal{A}_{\pi'}(w, w')); (w' \models \rho)\} \\
&= \quad \{ \text{ (7)} \} \\
&\quad \sum_{w' \in W} \{(\mathcal{A}_{\pi}(w, w'); (w' \models \rho) + \mathcal{A}_{\pi'}(w, w'); (w' \models \rho))\} \\
&= \quad \{ \text{ lattice distributivity} \} \\
&\quad \sum_{w' \in W} \{(\mathcal{A}_{\pi}(w, w'); (w' \models \rho)\} + \sum_{w' \in W} \{\mathcal{A}_{\pi'}(w, w'); (w' \models \rho))\} \\
&= \quad \{ \text{ defn of } \models \} \\
&\quad (w \models \langle\pi\rangle\rho) + (w \models \langle\pi'\rangle\rho) \\
&= \quad \{ \text{ defn of } \models \} \\
&\quad (w \models \langle\pi\rangle\rho \vee \langle\pi'\rangle\rho)
\end{aligned}
$$

Therefore $\langle\pi + \pi'\rangle\rho \leftrightarrow \langle\pi\rangle\rho \vee \langle\pi'\rangle\rho$ is valid.

Axiom (2.2)

$$
\begin{aligned}
& (w \models \langle\pi\rangle\langle\pi'\rangle\rho) \\
= \quad & \{ \text{ defn of } \models\} \\
& \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w \models \langle\pi'\rangle\rho)\} \\
= \quad & \{ \text{ defn of } \models\} \\
& \sum_{w' \in W} \big\{\mathcal{A}_\pi(w, w'); \sum_{w'' \in W} \{\mathcal{A}_{\pi'}(w', w''); (w'' \models \rho)\}\big\} \\
= \quad & \{ (7)\} \\
& \sum_{w' \in W} \big\{ \sum_{w'' \in W} \{\mathcal{A}_\pi(w, w'); \mathcal{A}_{\pi'}(w', w''); (w'' \models \rho)\}\big\} \\
= \quad & \{ \text{commutativity}\} \\
& \sum_{w'' \in W} \big\{ \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); \mathcal{A}_{\pi'}(w', w''); (w'' \models \rho)\}\big\} \\
= \quad & \{ \text{ since } (w'' \models \rho) \text{ is independent of } w'\} \\
& \sum_{w'' \in W} \big\{ \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); \mathcal{A}_{\pi'}(w', w'')\}; (w'' \models \rho)\big\} \\
= \quad & \{ \text{defn. of composition}\} \\
& \sum_{w'' \in W} \big\{\mathcal{A}_{\pi;\pi'}(w, w''); (w'' \models \rho)\big\} \\
= \quad & \{ \text{ defn. of } \models\} \\
& (w \models \langle\pi;\pi'\rangle\rho)
\end{aligned}
$$

Therefore $\langle\pi\rangle\langle\pi'\rangle\rho \leftrightarrow \langle\pi;\pi'\rangle\rho$ is valid.

Axiom (2.3)

$$
\begin{aligned}
& (w \models \langle\pi\rangle\bot) \\
= \quad & \{ \text{ defn. of } \models\} \\
& \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); (w \models \bot)\} \\
= \quad & \{ \text{defn. of satisfaction}\} \\
& \sum_{w' \in W} \{\mathcal{A}_\pi(w, w'); \bot\} \\
= \quad & \{ (9) \text{ and } \bot = 0\}
\end{aligned}
$$

$$\begin{aligned} & \sum_{w' \in W} \{\bot\} \\ = \quad & \{ \ (4)\} \\ & \bot \end{aligned}$$

Therefore $\langle\pi\rangle 0 \leftrightarrow 0$ is valid.

Lemma 3. *The following are valid formulas in any $\mathcal{GDL}(\mathbf{A})$:*

(3.1) $\langle\pi\rangle\rho \rightarrow \langle\pi^*\rangle\rho$
(3.2) $\langle\pi^*\rangle\rho \leftrightarrow \langle\pi^*;\pi^*\rangle\rho$
(3.3) $\langle\pi^*\rangle\rho \leftrightarrow \langle\pi^{**}\rangle\rho$
(3.4) $\langle\pi^*\rangle\rho \leftrightarrow \rho \vee \langle\pi\rangle\langle\pi^*\rangle\rho$

Proof. **Axiom (3.1)** In order to proof this axiom we have first to observe that for any $a, b, c \in A$, $a \leq b$ implies $a;c \leq b;c$. Supposing $a \leq b$, i.e., $a + b = b$, we have that

$$a;c + b;c \ =_{\{(8)\}} \ (a + b);c \ =_{\{\text{by hypothesis } a+b=b\}} \ b;c$$

i.e., $a;c \leq b;c$. Moreover, we have also to check that $a \leq a^*$ which comes directly from (10) by monotonicity of the supremum and transitivity. Hence (and since $\mathbb{M}_n(\mathbf{A})$ is an action lattice), we have for any $w \in W$,

$$\begin{aligned} & \mathcal{A}_\pi(w, w') \leq \mathcal{A}_{\pi^*}(w, w') \text{ for any } w' \in W \\ \Rightarrow \quad & \{ \ a \leq b \text{ implies } a;c \leq b;c\} \\ & \mathcal{A}_\pi(w, w');(w' \models \rho) \leq \mathcal{A}_{\pi^*}(w, w');(w' \models \rho) \text{ for any } w' \in W \\ \Rightarrow \quad & \{ \text{ monotonicity of the supremum}\} \\ & \sum_{w' \in W} \{\mathcal{A}_\pi(w, w');(w' \models \rho)\} \leq \sum_{w' \in W} \{\mathcal{A}_{\pi^*}(w, w');(w' \models \rho)\} \\ \Leftrightarrow \quad & \{ \text{ defn of } \models\} \\ & (w \models \langle\pi\rangle\rho) \leq (w \models \langle\pi^*\rangle\rho) \\ \Leftrightarrow \quad & \{ \text{ defn of } \models\} \\ & (w \models \langle\pi\rangle\rho \rightarrow \langle\pi^*\rangle\rho) \end{aligned}$$

Therefore $\langle\pi\rangle\rho \rightarrow \langle\pi^*\rangle\rho$ is valid.

Axioms (3.2),(3.3) and (3.4) We start recalling the following well known Kleene algebra properties: $a^* = a^{**}$, $a^* = a^*;a^*$ and $1 + a;a^* = a^*$ (see [9]). Therefore, we have that

$$\mathcal{A}_{\pi^*}(w, w') = \mathcal{A}_{\pi^{**}}(w, w') \tag{23}$$
$$\mathcal{A}_{\pi^*}(w, w') = \mathcal{A}_{\pi^*;\pi^*}(w, w') \tag{24}$$
$$\mathcal{A}_{1+\pi;\pi^*}(w, w') = \mathcal{A}_{\pi^*}(w, w') \tag{25}$$

The remaining of the first two proofs follows exactly the same steps of the one for Axiom **(3.1)**. For the third case, we have that for any $w \in W$,

$$
\begin{array}{ll}
& \mathcal{A}_{1+\pi;\pi^*}(w,w') = \mathcal{A}_{\pi^*}(w,w') \text{ for any } w' \in W \\
\Leftrightarrow & \quad \{ \text{ program interpretation} \} \\
& \mathcal{A}_1(w,w') + \mathcal{A}_{\pi;\pi^*}(w,w') = \mathcal{A}_{\pi^*}(w,w') \text{ for any } w' \in W \\
\Leftrightarrow & \quad \{ \ a = b \text{ iff } a;c = b;c \} \\
& \big(\mathcal{A}_1(w,w') + \mathcal{A}_{\pi;\pi^*}(w,w')\big);(w' \models \rho) = \mathcal{A}_{\pi^*}(w,w');(w' \models \rho) \\
& \text{for any } w' \in W \\
\Leftrightarrow & \quad \{ \ (7) \} \\
& \mathcal{A}_1(w,w');(w' \models \rho) + \mathcal{A}_{\pi;\pi^*}(w,w');(w' \models \rho) = \mathcal{A}_{\pi^*}(w,w');(w' \models \rho) \\
& \text{for any } w' \in W \\
\Rightarrow & \quad \{ \text{ supremum funcionality} \} \\
& \sum_{w' \in W} \{\mathcal{A}_1(w,w');(w' \models \rho) + \mathcal{A}_{\pi;\pi^*}(w,w');(w' \models \rho)\} = \\
& \sum_{w' \in W} \{\mathcal{A}_{\pi^*}(w,w');(w' \models \rho)\} \\
\Leftrightarrow & \quad \{ \text{ distributivity} \} \\
& \sum_{w' \in W} \{\mathcal{A}_1(w,w');(w' \models \rho)\} + \sum_{w' \in W} \{\mathcal{A}_{\pi;\pi^*}(w,w');(w' \models \rho)\} = \\
& \sum_{w' \in W} \{\mathcal{A}_{\pi^*}(w,w');(w' \models \rho)\} \\
\Leftrightarrow & \quad \{ \ \textstyle\sum_{w' \in W} \{\mathcal{A}_1(w,w');(w' \models \rho) = (w \models \rho)\} + \text{program interpretation} \} \\
& (w \models \rho) + (w \models \langle \pi;\pi^* \rangle \rho) = (w \models \langle \pi^* \rangle \rho) \\
\Leftrightarrow & \quad \{ \ \textbf{(2.2)} \} \\
& (w \models \rho) + (w \models \langle \pi \rangle \langle \pi^* \rangle \rho) = (w \models \langle \pi^* \rangle \rho) \\
\Leftrightarrow & \quad \{ \text{ defn of } \models \} \\
& (w \models \rho \vee \langle \pi \rangle \langle \pi^* \rangle \rho) = (w \models \langle \pi^* \rangle \rho)
\end{array}
$$

Therefore, $\langle \pi^* \rangle \rho \leftrightarrow \rho \vee \langle \pi \rangle \langle \pi^* \rangle \rho$ is valid.

4 Conclusions

The method introduced in this paper is able to generate several dynamic logics useful for the working Software Engineer. Some of them are documented in

the literature, others freshly new. For instance, for verification of imperative programs, we may consider a logic whose states are valuations of program variables. Hence, and as usual, atomic programs become assignments of variables. In this context, a transition $w \rightarrow^{x:=a} w'$ means that the state w' differs from w just in the value of variable x, i.e., that $w'(x) = a$ and for any variable $y \neq x$, $w'(y) = w(y)$.

A very natural direction for future work is to enrich this framework with tests, i.e., programs $?cond$ interpreted as $\mathcal{A}_{?cond} = \{(w, w) | w \models cond\}$. As usual, this provides a way to express *if-then-else* statements in dynamic logics. Another topic deserving attention is the characterisation of program refinement in this setting, witnessed by some class of action lattice morphisms.

Acknowledgements. This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within projects FCOMP-01-0124-FEDER-037281, PEst-OE/MAT/UI4106/2014, FCOMP-01-0124-FEDER-028923 and by project NORTE-07- 0124-FEDER-000060, co-financed by the North Portugal Regional Operational Programme (ON.2), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

References

1. B. Beckert. A dynamic logic for the formal verification of java card programs. In I. Attali and T. P. Jensen, editors, *Java Card Workshop*, volume 2041 of *Lecture Notes in Computer Science*, pages 6–24. Springer, 2000.
2. M. Droste and P. Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007.
3. H. Furusawa. The categories of kleene algebras, action algebras and action lattices are related by adjunctions. In R. Berghammer, B. M oller, and G. Struth, editors, *RelMiCS*, volume 3051 of *Lecture Notes in Computer Science*, pages 124–136. Springer, 2003.
4. S. F. Goble. Grades of modality. *Logique et Analyse*, 13:323–334, 1970.
5. S. Gottwald. *A Treatise on Many-Valued Logics*. Studies in Logic and Computation (volume 9). Research Studies Press, 2001.
6. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
7. D. Kozen. On action algebras. manuscript in: Logic and Flow of Information, Amsterdam, 1991.
8. D. Kozen. A probabilistic PDL. *J. Comput. Syst. Sci.*, 30(2):162–178, 1985.
9. D. Kozen. A completeness theorem for kleene algebras and the algebra of regular events. *Inf. Comput.*, 110(2):366–390, 1994.
10. B. Lopes, M. R. F. Benevides, and E. H. Haeusler. Propositional dynamic logic for petri nets. *Logic Journal of the IGPL*, 22(5):721–736, 2014.
11. O. Mürk, D. Larsson, and R. Hähnle. Key-c: A tool for verification of c programs. In F. Pfenning, editor, *CADE*, volume 4603 of *Lecture Notes in Computer Science*, pages 385–390. Springer, 2007.
12. N. J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28(1):71–87, 1986.

13. A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.
14. A. Platzer. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Logical Methods in Computer Science*, 8(4), 2012.
15. V. R. Pratt. Action logic and pure induction. In *JELIA*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120. Springer, 1990.
16. W. van der Hoek. On the semantics of graded modalities. *Journal of Applied Non-Classical Logics*, 2(1), 1992.